基于 GPU 加速遗传算法的直接定位研究 *

任衍青，逯志宇，王大鸣

(信息工程大学 信息系统工程学院，郑州 450001)

摘　要：针对大规模数据下遗传直接定位算法执行时间慢、实时性较差问题，提出了基于 GPU 加速的并行遗传直接定位算法。根据直接定位代价函数特点，设计了 GPU 高速并行遗传进化架构，通过对适应度函数并行化计算以及对选择、交叉、变异等遗传操作并行化设计，缩短了算法执行时间，提高了算法执行效率。仿真实验表明，通过合理的 GPU 并行线程结构设计，显著提升了遗传直接定位算法的执行速度，可更快得到直接定位估计结果。

关键词：直接定位；GPU 加速；遗传算法

中图分类号：TN911.72　　doi: 10.3969/j.issn.1001-3695.2017.11.0746

Research on direct position determination based on GPU accelerated genetic algorithm

Ren Yanqing, Lu Zhiyu, Wang Daming

(*Institute of Information System Engineering*, *Information Engineering University Zhengzhou* 450001, *China*)

Abstract: The genetic direct position determination (DPD) algorithm executes slowly and has poor real-time performance under the large-scale data condition. This paper proposed a GPU-based genetic DPD algorithm to overcome the above shortcoming. According to the cost function of DPD, it designed a high speed parallel architecture of GPU. It reduced the execution time and improved efficiency, according to the parallel design of the fitness function and the genetic operation such as selection, crossover and mutation. The experiments show that, through reasonable design of the parallel thread architecture of GPU, the proposed method can reduce the execution time of the genetic DPD algorithm efficiently and locate the emitter faster.

Key words: direct position determination; GPU accelerated; genetic algorithm

0　引言

传统的无源定位体制一般采用两步定位法，即首先估计出目标的定位参量，然后通过定位方程实现目标位置解算。两步定位方法虽然处理过程较为简单，但由于参数估计和位置解算相分离，不可避免地造成数据处理过程中位置信息的损失，因而无法获得最优的估计性能。直接定位技术（direct position determination，DPD）避免了两步定位方法因两步分离造成的信息损失，因而具有更高的定位精度，已成为近年来无线定位领域研究热点[1~4]。

直接定位技术虽然具有更高的定位精度，但其计算复杂度也明显增加。现阶段的直接定位方法几乎均基于遍历思想，采用网格搜索方式实现对目标位置的估计。其计算复杂度随着网格搜索范围和搜索精细度增加而呈几何级数增加，给该类方法的实际应用造成了困难。近年来兴起的智能优化算法[5]为解决直接定位搜索量大问题提供了新思路。文献[6]利用遗传算法（genetic algorithm，GA）的概率传递规则代替确定性规则，避免搜索大量无意义点，有效提升了直接定位算法的搜索效率。然而随着网格搜索范围的增加，必然要求通过增加种群规模来减小算法陷入局部最优的可能性。遗传算法在种群数量和观测数据规模增加的情况下计算量仍然十分巨大，搜索时间长、实时性差，阻碍了遗传算法在直接定位中的有效应用。

遗传算法中个体之间相互独立，具有天然的并行特性，并行化设计是提升算法速度的主要途径。目前遗传算法大多基于 CPU 多线程设计[6,7]，由于 CPU 的硬件结构特点所决定，CPU 线程数较少，并不适合做大规模并行计算，基于 CPU 的遗传算法对直接定位搜索的速度提升有限。近年来基于图形处理器（graphics processing unit，GPU）的并行计算研究引起学者广泛关注[8~11]。GPU 的内核资源丰富，具有大规模甚至超大规模的多线程并行计算能力。通过结合遗传算法个体之间独立性与 GPU 并行化的硬件结构特点，可有效加快遗传算法迭代速度。但目前 GPU 遗传算法主要集中于种群个体并行性[12]、随机数产生[13,14]等方面的并行化研究，并未针对不同适应度函数的特点进行并行优化设计。因此传统的 GPU 遗传算法并不完全适

收稿日期：2017-11-20；**修回日期：**2017-12-28　**基金项目：**国家高技术研究发展计划资助项目（2012AA01A502，2012AA01A505）；国家自然科学基金资助项目（61401513）

作者简介：任衍青（1992-），男，山东枣庄人，硕士研究生，主要研究方向为直接定位、智能优化算法（yq_renice@163.com）；逯志宇（1989-），男，博士研究生，通信信号处理、智能计算；王大鸣（1971-），男，教授，博士，主要研究方向为卫星移动通信、导航与定位.

合在直接定位中应用，加速性能有待进一步提升。

综上所述，在大规模种群条件下，基于 CPU 的遗传直接定位算法（GA-DPD）迭代时间较长、实时性较差；而目前经过 GPU 加速的遗传算法用于对目标的直接定位过程中，未能针对适应度函数的特点作进一步并行优化设计，仍有进一步加速的空间。本文首先针对直接定位代价函数的特点合理设计并行化适应度函数计算方法；然后并行化选择、交叉、变异等操作，最大程度地将 GA-DPD 算法的各部分进行并行化处理，提高算法的迭代速度。实验部分分别将 GA-DPD 算法与本文所设计的 GPU 并行遗传直接定位算法（GPU-GADPD）进行对比，证明本文方法实现进一步加速的有效性。

1 问题模型

1.1 直接定位模型

假设存在一位置为 $\boldsymbol{p}_0=[x_0,y_0]^{\mathrm{T}}$ 的静止目标信号源，发射载频为 f_c 的窄带信号 $s(t)$。L 个运动观测站在 K 个观测间隙内对目标信号进行观测，则在 t 时刻的观测数据模型为

$$r_{l,k}(t)=b_{l,k}s_k(t)e^{j2\pi f_{l,k}t}+n_{l,k}(t),0\le t\le T \tag{1}$$

其中：$b_{l,k}$ 表示信号在第 k 个观测间隙到达第 l 个观测站的复传播系数；$s_k(t)$ 表示信号在第 k 个观测间隙内的复包络；$n_{l,k}(t)$ 表示高斯白噪声；T 表示观测间隙长度；$f_{l,k}$ 表示第 l 个观测站在第 k 个观测间隙观测到的信号频率，并有

$$f_{l,k}=f_c(1+\mu_{l,k}) \tag{2}$$

其中：$\mu_{l,k}$ 表示由于目标和观测站相对位移引起的多普勒效应。经过数字下变频后，对每个观测间隙内的信号进行 N 快拍采样，即采样间隔为 $T_s=\dfrac{T}{N-1}$，观测矢量可表示为

$$\boldsymbol{r}_{l,k}=b_{l,k}\boldsymbol{A}_{l,k}\boldsymbol{s}_k+\boldsymbol{n}_{l,k} \tag{3}$$

$$\begin{cases}\boldsymbol{r}_{l,k}\triangleq[r_{l,k}[0],r_{l,k}[1],\cdots,r_{l,k}[N-1]]^{\mathrm{T}}\\ \boldsymbol{n}_{l,k}\triangleq[n_{l,k}[0],n_{l,k}[1],\cdots,n_{l,k}[N-1]]^{\mathrm{T}}\\ \boldsymbol{s}_k\triangleq[s_k[0],s_k[1],\cdots,s_k[N-1]]^{\mathrm{T}}\\ \boldsymbol{A}_{l,k}\triangleq\mathrm{diag}\{1,e^{j2\pi f_c\mu_{l,k}T_s},\cdots,e^{j2\pi f_c\mu_{l,k}(N-1)T_s}\}\end{cases} \tag{4}$$

其中：$\mathrm{diag}\{*\}$ 表示矢量对角化操作；$\boldsymbol{r}_{l,k}$ 和 $\boldsymbol{s}_k$ 为 $N\times1$ 维矢量；$\boldsymbol{A}_{l,k}$ 为 $N\times N$ 维矩阵。

在此不加证明地给出，式(1)直接定位模型的最大似然代价函数[15]为

$$L=\sum_{k=1}^{K}\sum_{l=1}^{L}\left|(\boldsymbol{A}_{l,k}\boldsymbol{s}_k)^{\mathrm{H}}\boldsymbol{r}_{l,k}\right|^2 \tag{5}$$

故该直接定位模型的最大似然估计为

$$\hat{\boldsymbol{p}}_0=\underset{\boldsymbol{p}}{\mathrm{argmax}}\{L\} \tag{6}$$

1.2 遗传直接定位算法

GA-DPD 算法的主要流程如图 1 所示。初始种群由位置坐标 $[x,y]^{\mathrm{T}}$ 构成，这些位置坐标在搜索范围内随机产生。适应度函数用于描述个体对环境的适应能力，适应度函数值越大，说明个体适应能力越强，个体越接近问题的最优解[5]。此处以式(5)直接定位代价函数值作为个体的适应度函数值。选择操作是从当前种群中选择适应度值较高的个体作为进行下步遗传操作的父本，常采用正比选择策略，即个体被选择的概率为该个体适应度函数值占种群所有个体适应度函数值总和的比例。假设第 i 个个体的适应度函数值为 F_i，种群数量为 P_s，则该个体被选中进行选择操作的概率为

$$P_i=\frac{F_i}{\sum_{i=1}^{P_s}F_i} \tag{7}$$

得到该个体的选择概率后，采样“轮盘赌”方式决定是否进行选择操作。交叉操作是以给定的交叉概率决定父本个体间是否相互进行杂交计算产生两个新的个体。变异操作是以给定的变异概率决定父本个体是否进行变异运算产生全新个体。

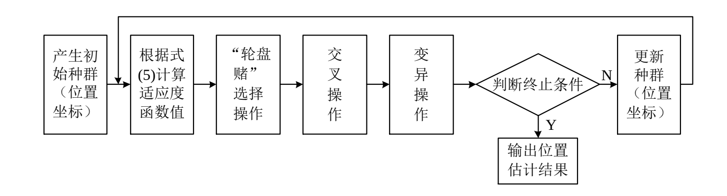


图 1 GA-DPD 算法基本流程

综上所述，通过式(5)可以发现，完成一个网格搜索，共需要进行 $KL(N^2+N)$ 次乘法运算。假设采样长度 $N=10^6$ 且 $K\times L=6$，种群数 $P_s=1000$，完成一次遗传迭代计算适应度函数所需的乘法次数为 60 亿次。可以看出，虽然遗传算法采用智能搜索方式有效改善了遍历搜索所造成的巨大计算压力，但随着种群数目扩大和数据规模增加，迭代过程中的计算量急剧增加、计算耗时加大、实时性变差，给算法的实际应用带来困难。接下来通过对 GA-DPD 算法进行 GPU 加速并行化设计，提升算法的计算效率，缩短算法迭代时间，以期满足工程应用需求。

2 基于 GPU 加速的并行遗传直接定位算法

2.1 适应度函数并行化

通过第 1 章可以看到，GA-DPD 算法的适应度函数计算中需要进行与采样点相关的乘法操作，当采样点数较大时，适应度函数的计算将耗费大量时间，成为算法效率提升的主要瓶颈。若能对适应度函数的计算进行并行处理，可大大提升算法的执行效率。GPU 中线程并行化设计体现在线程结构的组织上，GPU 的基本编程模型[16]如图 2 所示，包含线程块（Block）层和线程（Thread）层两个并行逻辑层。其中一个 Block 中包含 G 个 Thread。不同的 Block 间相互并行，同一个 Block 中的 Thread 间也相互并行。

由于一个 Block 中的最大线程数量（一般 $G=1024$）可能远低于 GA-DPD 算法中采样点的数量，若仅采用一个 Block 中的线程处理 GA-DPD 所有采样点，意味着每个线程将分担处理多个采样点运算，而其余 Block 处于空闲状态，对资源造成浪费，

导致算法的并行效率低下。最大化线程数量是使用 GPU 的一个重要优化策略[10]，一个 Block 显然无法发挥 GPU 的真实性能，故采用如下方式使每个线程处理单个采样点，使调动的线程数量最大化。

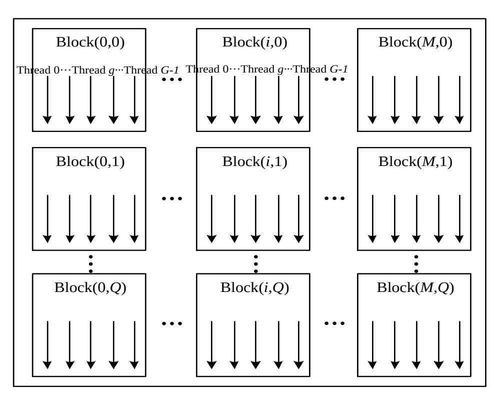


图 2 GPU 基本编程模型

首先根据种群数量设置共 P_s 行 Block 阵列，并假设每个 Block 中调动的线程数为 V，则每行共需调用 $D=\frac{N}{V}$ 个 Block。也即同一行的所有 Block 索引同一个个体，该行 Block 中被调用的线程分别处理该个体的一个采样点。以第 q 个个体计算适应度函数为例，线程分配情况如图 3 所示。

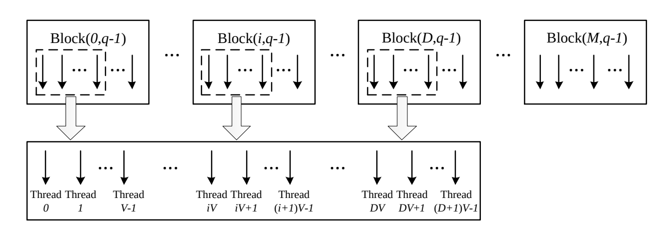


图 3 并行计算适应度函数线程分配示意图

图 3 表示的是第 $q-1$ 行的第 0 到第 D 个 Block 处理第 q 个个体所有数据，$\text{Thread}0\sim\text{Thread}(D+1)V-1$ 分别对应处理每一个采样点，通过此种方式调动最多的线程来计算适应度函数，使适应度函数的计算并行性得到最大化。此处应当说明的是，在进行线程分配时应尽可能使每个被调动的 Block 内的所有线程活跃，故 V 常设为 G，否则会造成 Block 中部分线程处于空闲状态，浪费计算资源。

2.2 遗传操作并行化

由 1.2 节 GA-DPD 算法流程可知，当得到种群的适应度函数值后，开始选择、交叉和变异遗传操作。由于种群中每个个体在选择、交叉和变异操作中相互独立，所以可对这些遗传操作作进一步 GPU 加速并行设计。对种群每个个体单独分配一个线程，具体线程分配方式如图 4 所示。对所有 Block 的线程统一编号分配，每个线程单独操作种群中一个个体，完成该个体的选择、交叉和变异操作，最大线程数等于种群数即 $BG+g=P_s$。

由 1.2 节式(7)可知，在进行选择操作之前需要计算种群所有个体的适应度函数值之和，可采用规约[16]思想对适应度求和进行 GPU 并行加速。由于一个线程对应处理一个个体的遗传操作，当种群个体数目超过一个 Block 的最大线程数 G 时，需要利用多个 Block 进行规约求和并行计算。但要注意每个 Block 内线程操作只能得到该 Block 内的线程所对应的适应度值求和中间结果，所有 Block 的中间结果构成一个中间结果数组，该数组共有 $\left\lceil \frac{P_s}{G} \right\rceil$ 个元素，$\lceil * \rceil$ 表示向上取整。需要对该数组进行二次规约才可得到种群所有个体的适应度之和。

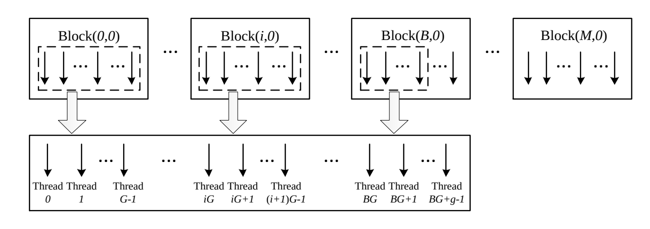


图 4 并行遗传操作线程分配示意图

在选择、交叉和变异遗传操作过程中，需要大量用到随机数组来决定子代种群的产生，而 GPU 并没有提供随机数生成接口，需要设计随机数生成机制。GPU 随机数生成算法得到广泛研究，比较成熟的主要有线性同余法和平方取中法。文献[13]分析了当产生随机数较多时，利用线性同余法产生的随机数组的均匀性比利用平方取中法产生的随机数组的均匀性更好。因此在本文所提出的 GPU-GADPD 算法中，采用线性同余法产生所需的随机数组。

2.3 GPU-GADPD 算法基本流程

利用 GPU 进行并行加速计算，需将原始数据从 CPU 内存拷贝到 GPU 显存中，待加速完成后，再将计算结果从显存拷贝到内存中。这种数据搬移会产生显著的通信开销，降低 GPU 加速的整体效率。所以在进行 GPU 并行化程序设计时，应尽量避免在内存与显存间进行不必要的数据迁移。基于上述原因，在 GPU-GADPD 算法设计过程中，除了观测数据和初始种群在内存中产生并向 GPU 显存搬移，计算种群适应度函数值、选择、交叉和变异操作均在 GPU 内执行，避免迭代数据在内存与显存间频繁传输，提高算法执行效率。GPU-GADPD 算法的数据迁移以及算法执行流程如图 5 所示。

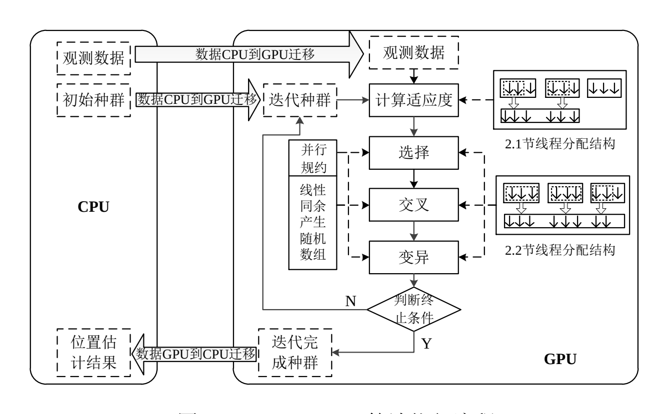


图 5 GPU-GADPD 算法执行流程

其中，针对适应度函数值和遗传操作的计算分别按照 2.1

和 2.2 节所设计的线程结构进行并行加速。从图 5 可以看出，在整个 GPU-GADPD 算法执行过程中，只有迭代开始前和结束后在 CPU 与 GPU 间进行了一次数据搬移，迭代过程始终在 GPU 内进行，减少了通信开销，提高了算法执行效率。

3　实验验证

仿真以 GPU-GADPD 算法完成一次迭代所耗费时间为衡量标准，验证所提算法的加速效果。考虑到每代种群迭代时间具有一定随机性，故设置最大迭代代数为 500 代，并取完成所有迭代的总时间对每一代的平均值作为完成一次迭代所耗费时间。假设目标位于 $(6.5,4)$km 处向外界发射载频为 0.2GHz 信号，两观测站速度大小为 $200\,\mathrm{m/s}$，分别在 $(1,0)$km、$(5,0)$km、$(9,0)$km 和 $(10,1)$km、$(10,5)$km、$(10,9)$km 处对目标进行观测。遗传参数设置为：交叉概率为 0.9，变异概率为 0.1。接下来将对 GA-DPD 算法和本文所提 GPU-GADPD 算法进行对比，在达到相同定位精度条件下，验证 GPU-GADPD 算法的加速效果。本文仿真实验软件参数配置如下：操作系统，Windows 7 64bit Ultimate；CUDA 开发平台，CUDA 7.0；程序运行平台，Visual Studio 2012 Ultimate。所用硬件参数配置如表 1 所示。

表 1　实验硬件参数配置

硬件	型号	主频	存储容量
CPU	Intel Core i7 6700K	4.0GHz	16G
GPU	NVIDIA GeForce GTX 970	1050MHz	4G

为了验证 GPU-GADPD 算法针对不同种群规模的加速效果，设置采样点数 $N=2^{10}$，得到随着种群规模增加两种算法的执行时间以及加速比结果，如表 2 所示，并根据表 2 得到图 6 所示加速效果。首先从表中可以看出，GA-DPD 算法的执行时间与种群规模基本呈线性关系，种群规模增加一倍，执行时间也增加约一倍，这是 GA-DPD 算法单线程执行方式所导致。而本文所提出的 GPU-GADPD 算法执行时间并不会随着种群规模增加而呈线性增加。只有当种群规模增加到足够大以后，算法执行时间才会随着种群规模翻倍而成倍增加。通过图 6 可以直观地看到，随着种群规模增加，所提算法的加速增益越来越明显，最高加速增益可达 40 倍以上，当种群规模增加到一定程度后，加速比值趋于稳定。因此本文所提出的基于 GPU 并行加速设计的 GPU-GADPD 算法能够带来较好的加速效果，证明了所提算法有效性。应当说明的是，限于实验条件限制，本文所采用的英伟达公司 GeForce 系列显卡属于普通级显卡，并不是做并行计算的专用显卡，若采用该公司的 Quadro 系列甚至 Tesla 系列高性能专业显卡，相信本文所提算法的加速效果将会更加明显。

为了进一步验证本文所提算法加速有效性，保持种群规模不变 $P_s=128$，针对不同采样长度，得到 GPU-GADPD 算法和 GA-DPD 算法执行时间实验结果，如表 3 所示，并由表 3 得到图 7 所示加速效果。与第一组实验结果一致，本文所提算法能够实现良好的加速效果。从图 6 可以看出，加速比变化呈现出先缓后快再缓的三段变化趋势。这可分别从三个方面解释这一现象：a）由于在算法执行过程中，GPU 需要不断从全局内存中取数据，当采样点数较少时，算法线程执行时间相对于内存访问时间微不足道，通过线程并行计算所得到加速增益不足以隐藏数据访问延迟，所以此阶段加速比值较小，加速效果不明显；b）随着采样点数增加，GPU 大规模并行计算优势逐渐凸显，能够较好地隐藏数据访问延迟，所以此阶段加速比明显提高；c）随着采样点数继续增加，GPU 趋于满负荷运行，达到了其最大计算能力，所以加速比不再随着数据规模的增大而显著增加，而呈现出平稳的趋势。通过以上分析可知，GPU 的优势在于对大规模数据的并行处理，通过调动尽可能多的线程执行并行计算弥补内存访问延迟，可获得更好的加速效果。

表 2　不同种群规模 GA-DPD 算法和 GPU-GADPD 算法计算时间对比

种群数	GA-DPD/ms	GPU-GADPD/ms	加速比
16	8.495	0.686	12.383
32	16.460	0.874	18.833
64	31.590	1.248	25.313
128	64.375	1.910	33.704
256	126.050	3.416	36.900
512	252.250	6.490	38.868
1024	505.440	12.512	40.396
2048	1009.710	24.906	40.541
4096	2040.450	49.358	41.340
8192	4095.955	98.592	41.545

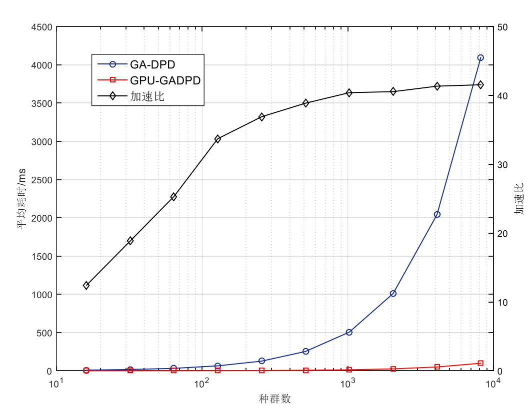


图 6　不同种群规模下 GPU-GADPD 算法加速效果

4　结束语

遗传算法在直接定位中的应用改善了传统直接定位算法遍历搜索效率低下问题，但当数据规模较大时，算法执行时间过长、实时性较差。针对此问题，本文提出了基于 GPU 加速的并行遗传直接定位算法。通过对适应度函数的并行化计算以及选择、交叉、变异等并行化遗传操作，显著缩短了算法执行时间。实验结果表明，利用本文所设计的线程分配结构对遗传直接定

位算法执行并行计算，可在现有 GPU 上获得 50 倍左右的加速效果，极大地缩短了算法计算时间，具有重要实际意义与工程应用价值。

表 3　不同采样长度 GA-DPD 算法和 GPU-GADPD 算法计算时间对比

采样点	GA-DPD/ms	GPU-GADPD/ms	加速比
2^6	4.680	1.590	2.943
2^7	8.735	2.202	3.967
2^8	16.615	3.682	4.513
2^9	32.135	5.272	6.095
2^{10}	63.725	1.904	33.469
2^{11}	125.030	3.120	40.074
2^{12}	250.380	5.460	45.857
2^{13}	501.615	10.014	50.091
2^{14}	1008.930	19.252	52.407
2^{15}	2029.640	37.720	53.808
2^{16}	4104.635	74.754	54.907
2^{17}	8192.625	148.902	55.020
2^{18}	16398.241	297.430	55.133

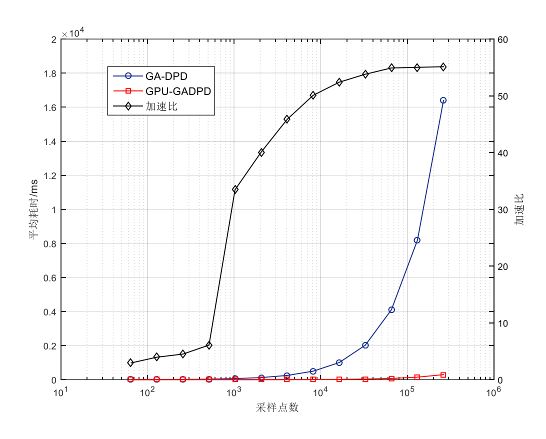


图 7　不同采样长度下 GPU-GADPD 算法加速效果

参考文献：

[1] 冯奇，曲长文，周强．多运动站异步观测条件下的直接定位算法 [J]. 电子与信息学报, 2017, 39 (2): 417-422.

[2] 逯志宇，任衍青，巴斌，等．基于分段信号相关累加的变速度多站联合直接定位方法 [J]. 物理学报, 2017, 66 (2): 66-75.

[3] Yin J X, Wu Y, Wang D. Direct position determination of multiple noncircular sources with a moving array [J]. Circuits Systems & Signal Processing, 2017, 36 (10): 4050-4076.

[4] Tzafri L, Weiss A. High resolution Direct position determination using MVDR [J]. IEEE Trans on Wireless Communications, 2016, 15 (9): 1-1.

[5] 汪定伟．智能优化方法 [M]. 北京：高等教育出版社, 2007.

[6] 任衍青，逯志宇，巴斌，等．锐化遗传直接定位快速估计算法 [J]. 西安电子科技大学学报：自然科学版, 2017, 44 (4): 144-150.

[7] 曹凯，陈国虎，江桦，等．自适应引导进化遗传算法 [J]. 电子与信息学报, 2014, 36 (8): 1884-1890.

[8] Tang M, Zhao J Y, Tong R F, *et al.* GPU accelerated convex hull computation [J]. Computers & Graphics, 2012, 36 (5): 498-506.

[9] Smith T M, Geijn R V D, Smelyanskiy M, *et al.* Anatomy of high-performance many-threaded matrix multiplication [C]// Proc of the 28th International Parallel and Distributed Processing Symposium. Phoenix: IEEE Press, 2014: 1049-1059.

[10] 方民权，张卫民，方建滨．GPU 编程与优化：大众高性能计算 [M]. 北京：清华大学出版社, 2016.

[11] Cheng L, Li T. Efficient data redistribution to speed up big data analytics in large systems [C]// Proc of the 23rd International Conference on High Performance Computing. Washington DC: IEEE Press, 2016: 99-100.

[12] 张庆科，杨波，王琳，等．基于 GPU 的现代并行优化算法 [J]. 计算机科学, 2012, 39 (4): 304-310.

[13] 谭彩凤，马安国，邢座程．基于 CUDA 平台的遗传算法并行实现研究 [J]. 计算机工程与科学, 2009, 31 (s1): 68-72.

[14] 刘欢，刘志勤，李凌，等．一种基于 CUDA 平台的随机数算法研究与实现 [J]. 计算机应用研究, 2017, 34 (5): 2727-2731.

[15] Amar A and Weiss A. Localization of narrowband radio emitters based on Doppler frequency shifts [J]. IEEE Trans on Signal Processing, 2008, 56 (11): 5500-5508.

[16] 张舒，褚艳利．GPU 高性能运算之 CUDA [M]. 北京：中国水利水电出版社, 2009.